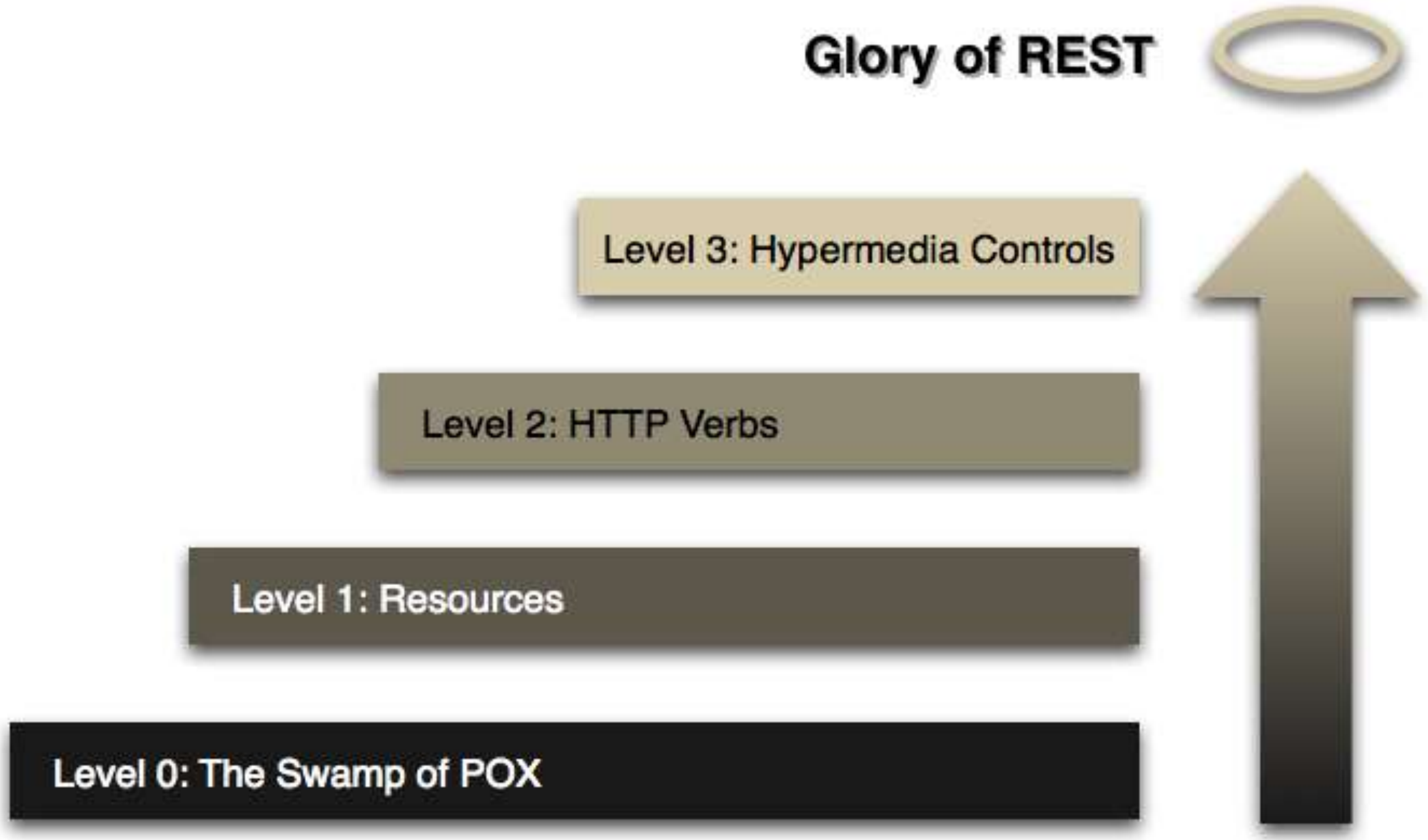
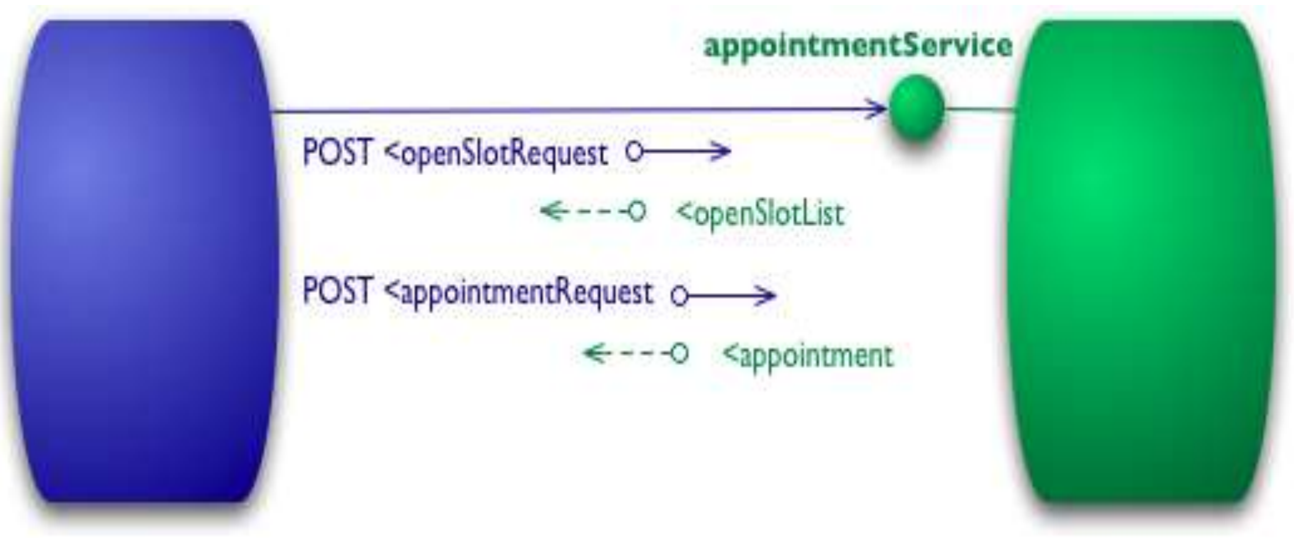
RESTFUL SERVICES

Richardson Maturity Model – *Martin Fowler*


Glory of REST
Level 3: Hypermedia Controls
Level 2: HTTP Verbs
Level 1: Resources
Level 0: The Swamp of POX

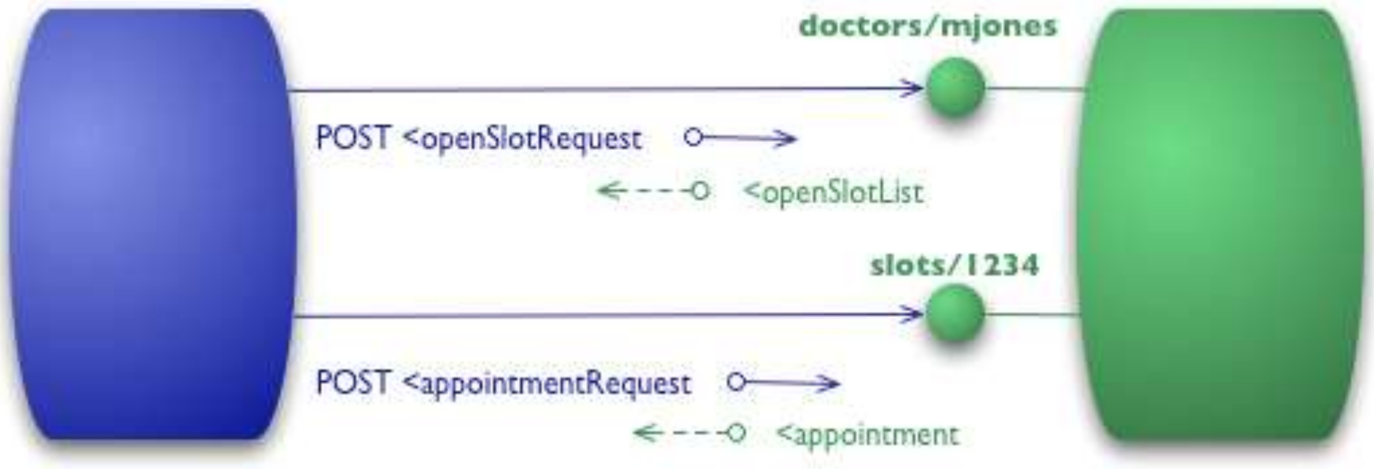
LEVEL 0

- HTTP as a transport system for remote interactions
- HTTP as a tunneling mechanism for your own Remote Procedure Invocation
- the content can actually be anything: JSON, YAML, key-value pairs, or any custom format



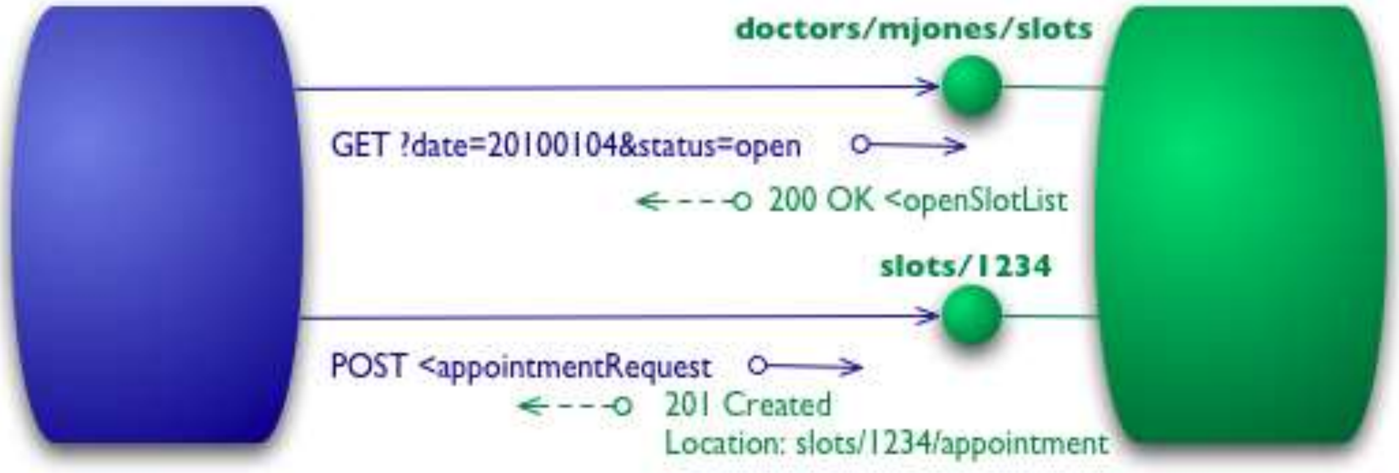
LEVEL 1

- The next step is the introduction of "resources"
- Instead of endpoints, we start communicating with resources
- Very similar to Object Oriented Programming Concepts



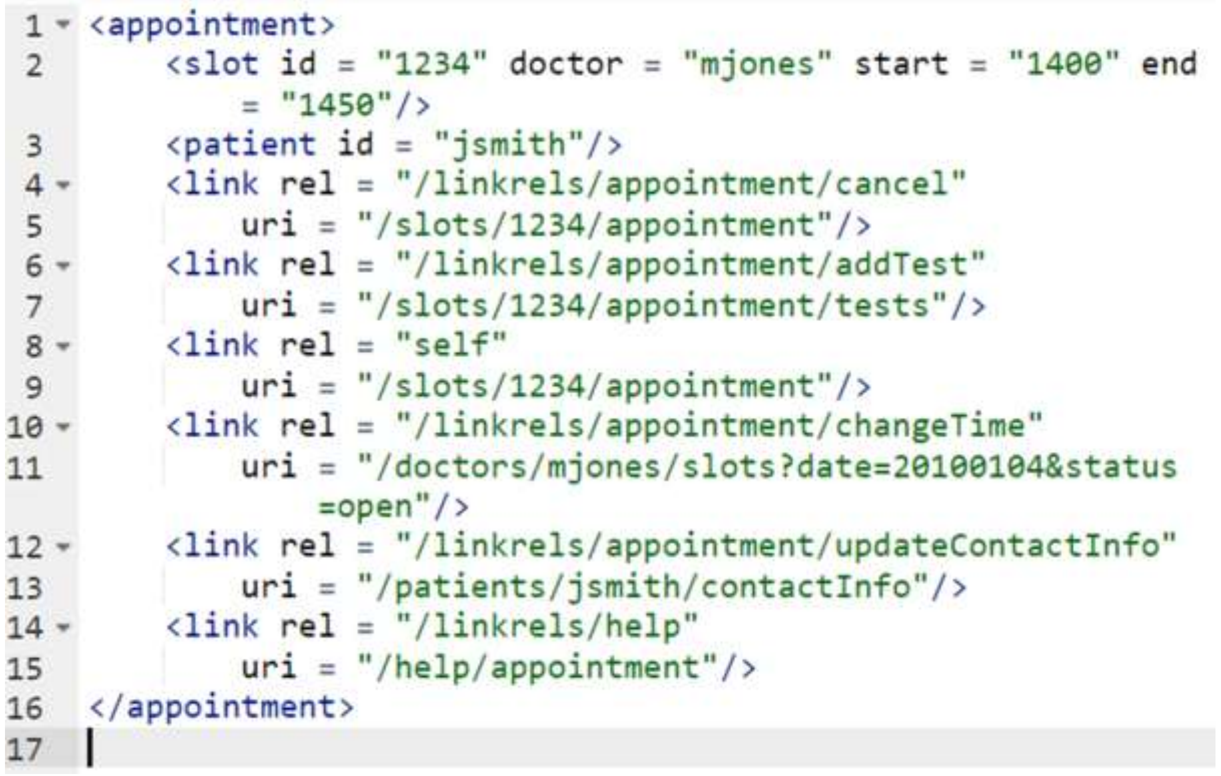
LEVEL 2

- Using the HTTP verbs as closely as possible to how they are used in HTTP itself
- Interpretation
  - GET (collection)
  - GET/id (individual)
  - POST (creation)
  - PATCH/id (update individual)
  - DELETE/id (delete individual)



LEVEL 3

- **HATEOAS** (Hypertext As The Engine Of Application State)
- Hypermedia controls is that they tell us what we can do next, and the URI of the resource we need to manipulate
- One obvious benefit of hypermedia controls is that it allows the server to change its URI scheme without breaking clients
- There's no absolute standard as to how to represent hypermedia controls. The 'REST in Practice' team, recommends ATOM (RFC 4287)



```
<appointment>
    <slot id = "1234" doctor = "mjones" start = "1400" end
        = "1450"/>
    <patient id = "jsmith"/>
    <link rel = "/linkrels/appointment/cancel"
        uri = "/slots/1234/appointment"/>
    <link rel = "/linkrels/appointment/addTest"
        uri = "/slots/1234/appointment/tests"/>
    <link rel = "self"
        uri = "/slots/1234/appointment"/>
    <link rel = "/linkrels/appointment/changeTime"
        uri = "/doctors/mjones/slots?date=20100104&status
            =open"/>
    <link rel = "/linkrels/appointment/updateContactInfo"
        uri = "/patients/jsmith/contactInfo"/>
    <link rel = "/linkrels/help"
        uri = "/help/appointment"/>
</appointment>
```

RESTFUL SERVICES

The Driver's view

“Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web”
- IBM

“ A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding. ”
- RedHat

GUIDELINES FOR RESTFUL SERVICE

1. Think **Resources** not End Points
   - There should be a way to uniquely identify each resource. URI is the de facto standard.
   - Has similarities to Object Oriented Design
2. **Uniform Interface**
   - Piggy back on HTTP Verbs (GET, POST, PATCH & DELETE) and also HTTP Response Codes
3. Decoupled **Representations**
   - Content can be accessed in variety of formats (json preferred for webservices)
4. **HATEOAS** - Hypermedia as the Engine of Application State
   - Hyperlink driven, explicit state transfer, stateless otherwise

GUIDELINES (EXTENDED)

1. **Client- Server Architecture**
   - RESTful style is mostly applicable (only) for ‘web’-services
2. **Stateless Communication**
   - No client information is stored between requests and each request is separate and unconnected
3. **Cacheable Data**
   - Cache wherever applicable to cutdown on roundtrips

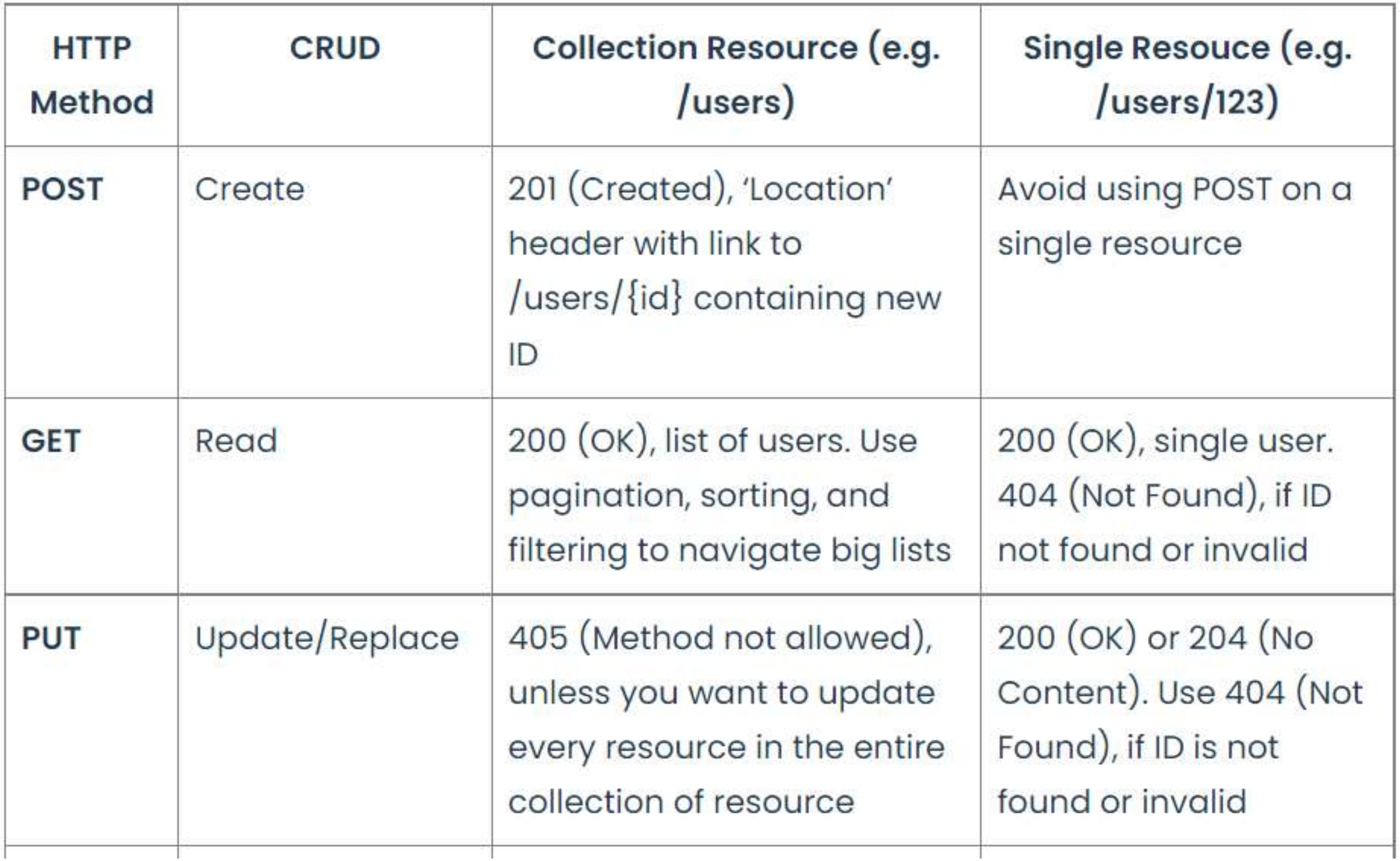
Note: Remember that there is no authority for styles. Once you are developer, prepare yourself for heated water cooler debates about what should and should not be included in the RESTful style..

GUIDELINES (EXTENDED)

1. **Client- Server Architecture**
   - RESTful style is mostly applicable (only) for ‘web’-services
2. **Stateless Communication**
   - No client information is stored between requests and each request is separate and unconnected
3. **Cacheable Data**
   - Cache wherever applicable to cutdown on roundtrips

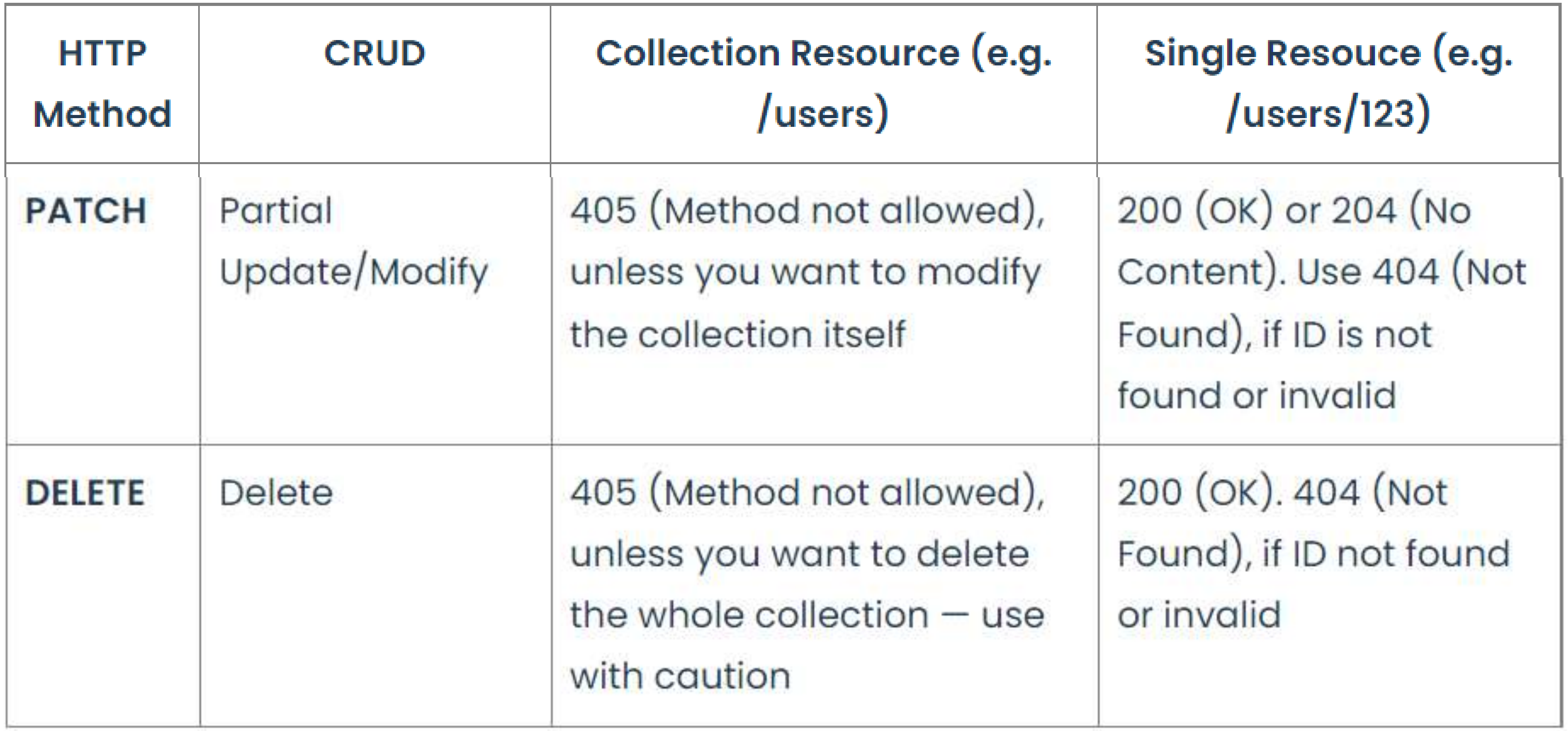
Note: Remember that there is no authority for styles. Once you are developer, prepare yourself for heated water cooler debates about what should and should not be included in the RESTful style..

HTTP METHODS/VERBS



HTTP Method	CRUD	Collection Resource (e.g. /users)	Single Resouce (e.g. /users/123)
POST	Create	201 (Created), 'Location' header with link to /users/{id} containing new ID	Avoid using POST on a single resource
GET	Read	200 (OK), list of users. Use pagination, sorting, and filtering to navigate big lists	200 (OK), single user. 404 (Not Found), if ID not found or invalid
PUT	Update/Replace	405 (Method not allowed), unless you want to update every resource in the entire collection of resource	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID is not found or invalid

HTTP METHODS/VERBS



HTTP Method	CRUD	Collection Resource (e.g. /users)	Single Resouce (e.g. /users/123)
PATCH	Partial Update/Modify	405 (Method not allowed), unless you want to modify the collection itself	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID is not found or invalid
DELETE	Delete	405 (Method not allowed), unless you want to delete the whole collection — use with caution	200 (OK). 404 (Not Found), if ID not found or invalid

HTTP RESPONSE CODES

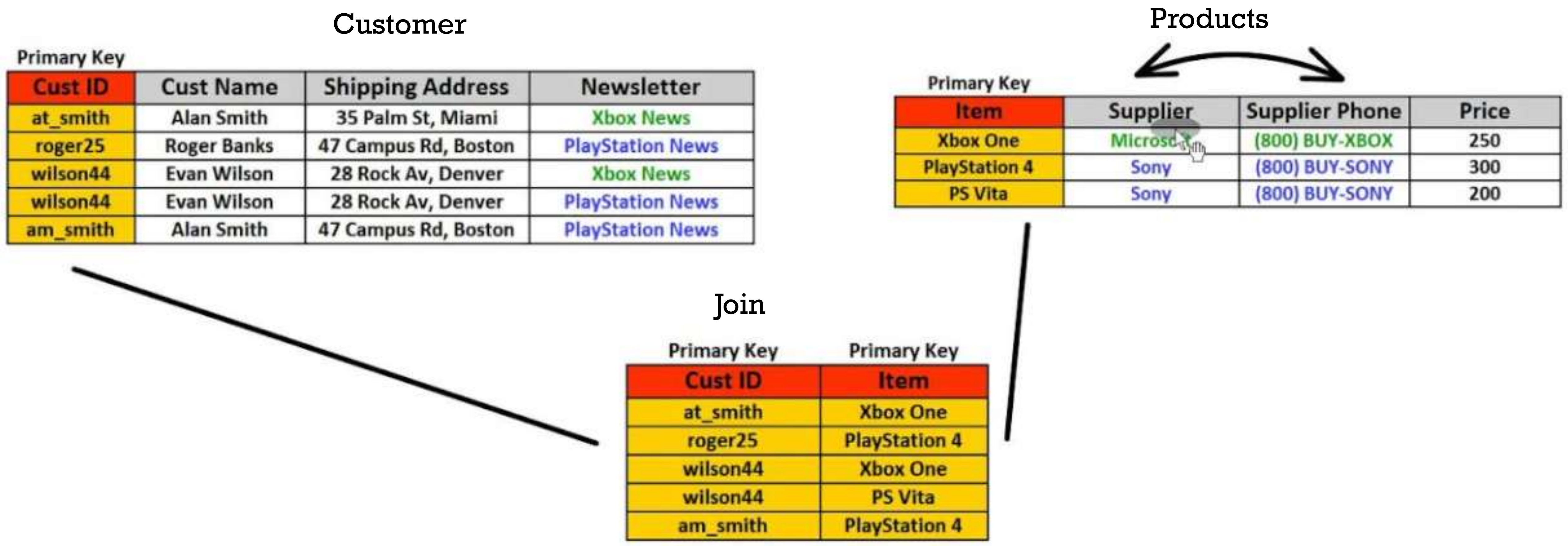
HTTP defines standard status codes that can be used to convey the results of a client's request. The status codes are divided into five categories.

1. 1xx: Informational – Communicates transfer protocol-level information.
2. 2xx: Success – Indicates that the client's request was accepted successfully.
3. 3xx: Redirection – Indicates that the client must take some additional action in order to complete their request.
4. 4xx: Client Error – This category of error status codes points the finger at clients.
5. 5xx: Server Error – The server takes responsibility for these error status codes.

https://restfulapi.net/http-status-codes/

HTTP RESPONSE CODES – IMPORTANT ONES

- **200 OK** Indicates that the request has succeeded.
- **201 Created** Indicates that the request has succeeded and a new resource has been created as a result.
- **400 Bad Request** The request could not be understood by the server due to incorrect syntax. The client SHOULD NOT repeat the request without modifications.
- **401 Unauthorized** Indicates that the request requires user authentication information. The client MAY repeat the request with a suitable Authorization header field
- **404 Not Found** The server can not find the requested resource.
- **500 Internal Server Error** The server encountered an unexpected condition that prevented it from fulfilling the request.



Customer

Primary Key

Cust ID	Cust Name	Shipping Address	Newsletter
at_smith	Alan Smith	35 Palm St, Miami	Xbox News
roger25	Roger Banks	47 Campus Rd, Boston	PlayStation News
wilson44	Evan Wilson	28 Rock Av, Denver	Xbox News
wilson44	Evan Wilson	28 Rock Av, Denver	PlayStation News
am_smith	Alan Smith	47 Campus Rd, Boston	PlayStation News

Products

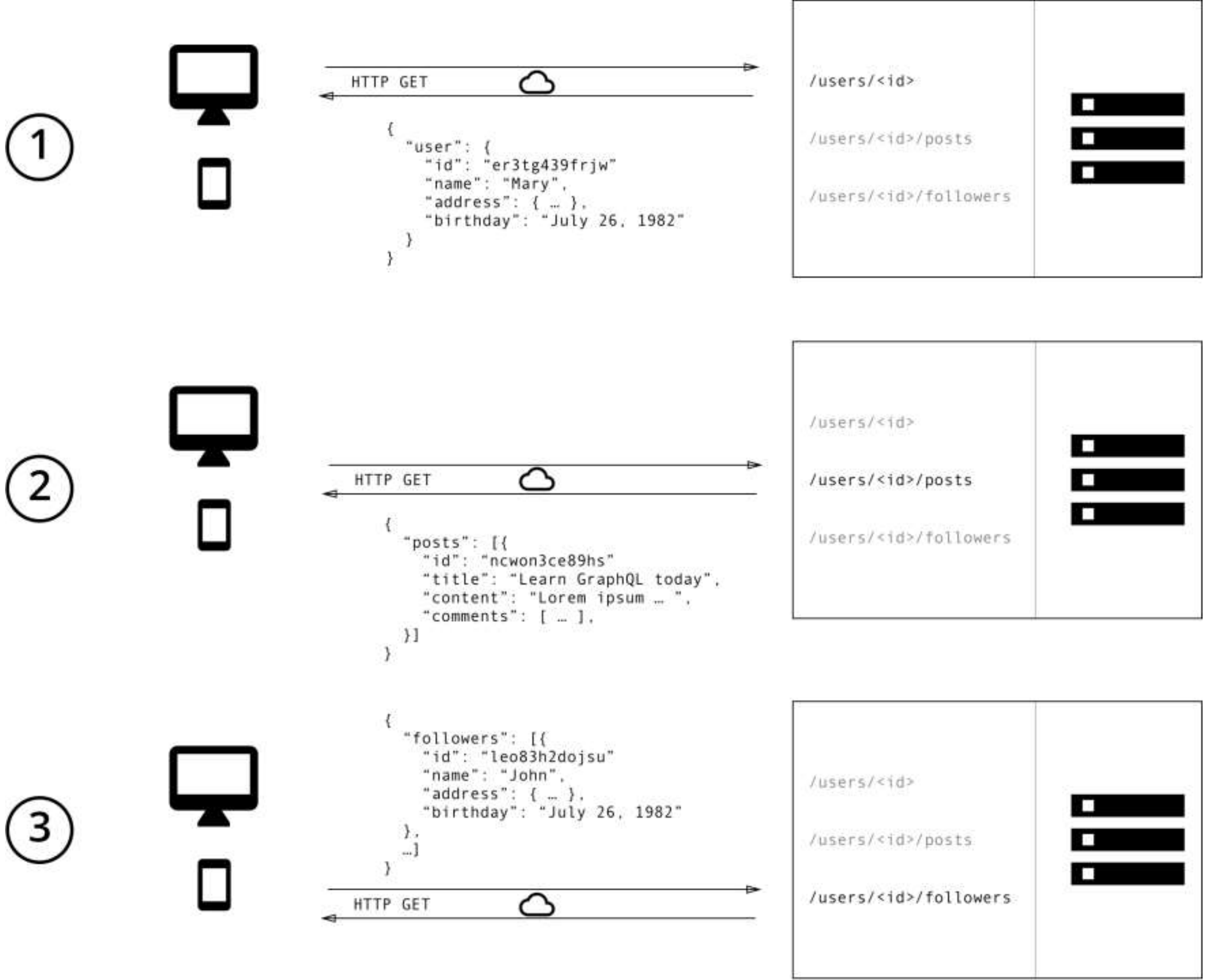
Primary Key

Item	Supplier	Supplier Phone	Price
Xbox One	Microsoft	(800) BUY-XBOX	250
PlayStation 4	Sony	(800) BUY-SONY	300
PS Vita	Sony	(800) BUY-SONY	200

Join

Primary Key / Primary Key

Cust ID	Item
at_smith	Xbox One
roger25	PlayStation 4
wilson44	Xbox One
wilson44	PS Vita
am_smith	PlayStation 4



```
{
  "user": {
    "id": "er3tg439frjw"
    "name": "Mary",
    "address": { … },
    "birthday": "July 26, 1982"
  }
}
```

```
{
  "posts": [{
    "id": "ncwon3ce89hs"
    "title": "Learn GraphQL today",
    "content": "Lorem ipsum … ",
    "comments": [ … ],
  }]
}
```

```
{
  "followers": [{
    "id": "leo83h2dojsu"
    "name": "John",
    "address": { … },
    "birthday": "July 26, 1982"
  },
  …]
}
```